Microsoft

Azure Developer Community Call

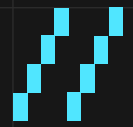
Cosmos DB Ask Me Anything
November 10th 2022

Contents

- How to design applications that are suited for Cosmos DB
- When Cosmos DB is not the right choice
- How to choose the right API and SDKs
- Global distribution vs. consistency model
- SLA and predicatable latency guarantees (what it really means)
- Elasticity and provision throughput vs. data modeling and query design

Code of Conduct

- Microsoft's mission is to empower every person and every organization on the planet to achieve more. This includes at this event, where we seek to create a respectful, friendly, and inclusive experience for all participants. As such, we do not tolerate harassing or disrespectful behavior, messages, images, or interactions by any event participant, in any form, at any aspect of the program including business and social activities, regardless of location.
- We do not tolerate any behavior that is degrading to any gender, race, sexual orientation or disability, or any behavior that would violate Microsoft's Anti-Harassment and Anti-Discrimination Policy, Equal Employment Opportunity Policy, or Standards of Business Conduct.
  In short, the entire experience must meet our culture standards.
- We encourage everyone to assist in creating a welcoming and safe environment. Please report any concerns, harassing behavior, suspicious or disruptive activity directly to us.
  Microsoft reserves the right to refuse admittance to or remove any person from this and/or future events at any time in its sole discretion.



CosmosDB Ask Me Anything

Microsoft

Azure Cosmos DB

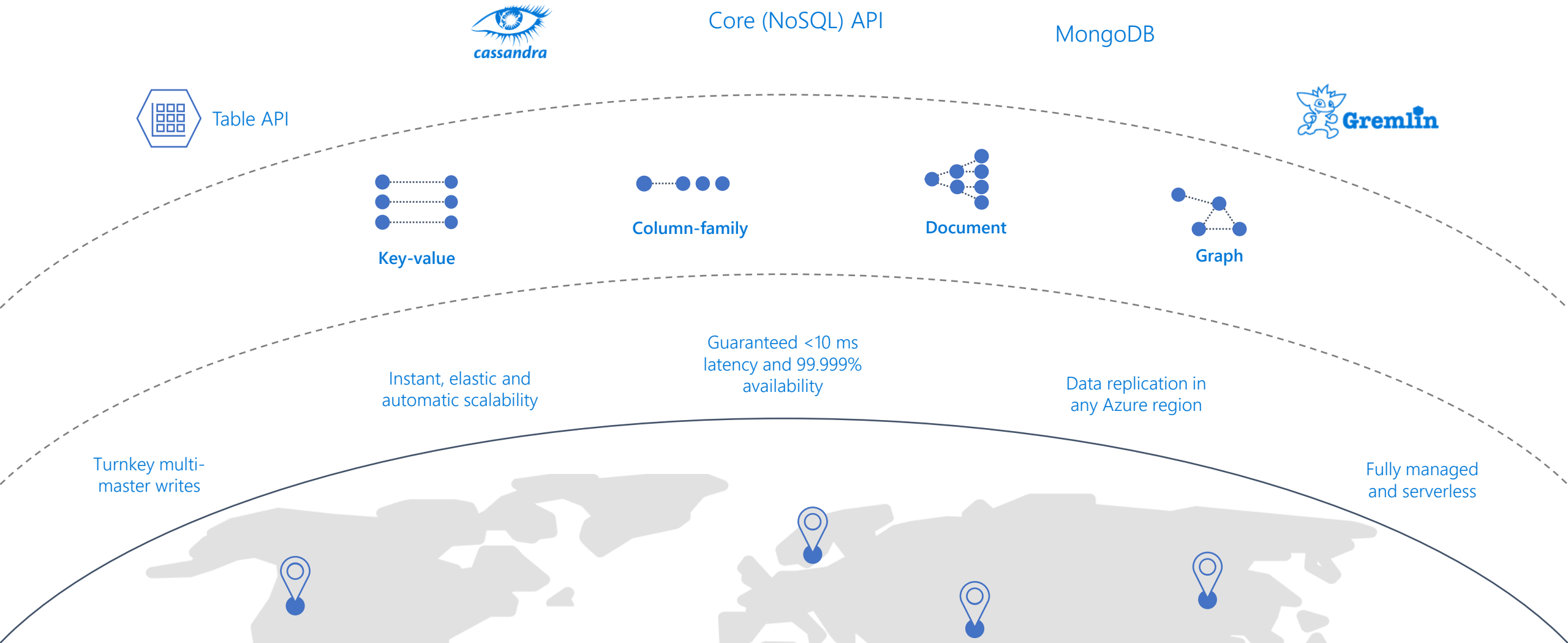
Ask Me Anything...

Fabian Meiswinkel – fabianm@microsoft.com

10.11.2022

AZURE COSMOS DB

Fast NoSQL database with open APIs for any scale

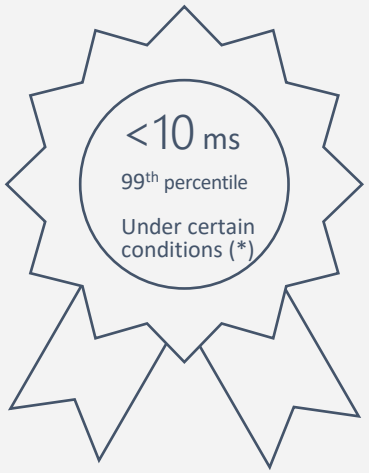


COMPREHENSIVE SLAs

RUN YOUR APP ON WORLD-CLASS INFRASTRUCTURE

Azure Cosmos DB is the only service with financially-backed SLAs for single-digit millisecond read and write latency at the 99th percentile, 99.999% high availability and guaranteed throughput and consistency

Latency



High Availability

Throughput

Consistency

(*) SQL Api, point operations for documents <= 1 KB, SDK with direct mode, compute client in same Azure region, Azure Accelerated Networking enabled, sufficient client-side compute capacity

SLA: https://azure.microsoft.com/en-us/support/legal/sla/cosmos-db/

PROVISIONED THROUGHPUT

- Expected throughput has to be **provisioned** (* except for Serverless)
- Expressed in **Request Units per second** (RU/s)
  - Represents the "cost" of a request in terms of CPU, memory and I/O
- Throughput can be provisioned:
  - at the database-level
  - at the container-level
- Provisioned throughput **can be changed programmatically** with API calls – and the decision how many RU/s to provision van be made manually or via auto-scale
- Design decisions like number of regions, single vs. multi-master, data model, consistency model, API choice can and will have impact on RU/s usage – so, implicitly cost
- More info:
  - https://learn.microsoft.com/en-us/azure/cosmos-db/how-to-choose-offer
  - https://learn.microsoft.com/en-us/azure/cosmos-db/throughput-serverless

How to design applications that are suited for Cosmos DB vs. when is it not the right choice?

- Sweet spot: Mission critical OLTP workloads
  - Relatively high percentage of point operations or queries targeting single logical partition
  - Geo distribution is needed
  - Latency sensitive
  - Elasticity/scale demanding
- Useful capabilities to extend applicable scenarios
  - Query engine/Indexing even when being schema-less
    - https://learn.microsoft.com/en-us/azure/cosmos-db/index-overview
    - https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/query/getting-started
  - Azure Synapse Link - https://learn.microsoft.com/en-us/azure/cosmos-db/synapse-link
- When Cosmos DB is not a good fit
  - Real-time analytics (requiring cross-partition queries over large data set (TBs or more)
    - RU/s are the virtual currency for CPU and I/O – real-time analytics scenarios often are very I/O hungry – which in Cosmos DB means the provisioned throughput (RU/s) in Cosmos for CPU and I/O is used simply for I/O – making I/O more expensive than alternatives
    - Better alternatives depending on workload can be Kusto/Data-Explorer, native Mongo (sharding tends to have fewer but larger shards, making aggregation pipeline more compelling in those scenarios) or Azure Sql and/or Postgres
  - Cold-storage scenarios – you have to provision at least 1 RU/GB of data stored. In cold-storage scenarios – PBs of data – very little throughput, this can also result in higher cost than when using Azure Storage (Table or Blob) for example

How to choose the right API and SDKs?

- APIs (decided on account creation, immutable, usually not interchangeable)
  - NoSql/Core – Api
    - All capabilities, best performance (only API covered under 10ms end-to-end latency SLA for point operations)
  - Mongo/Cassandra
    - OSS APIs – allowing to avoid vendor lock-in
    - Mongo and Cassandra APIs are not native Mongo or Cassandra - be careful with lift & shift
      - Mongo 4.2: https://learn.microsoft.com/en-us/azure/cosmos-db/mongodb/feature-support-42
      - Mongo 4.0: https://learn.microsoft.com/en-us/azure/cosmos-db/mongodb/feature-support-40
      - Mongo 3.6: https://learn.microsoft.com/en-us/azure/cosmos-db/mongodb/feature-support-36
      - Cassandra: https://learn.microsoft.com/en-us/azure/cosmos-db/cassandra/support
  - Table API
    - effectively a subset of NoSQL Api
    - only for migrations of existing apps using Azure Table Storage)
  - Gremlin API
    - Internally using NoSQL Api (Using NoSQL Api to query/write technically possible)
    - For graph-based queries

How to choose the right global distribution config and consistency model?

- https://learn.microsoft.com/en-us/azure/cosmos-db/distribute-data-globally
- https://learn.microsoft.com/en-us/azure/cosmos-db/high-availability
- https://learn.microsoft.com/en-us/azure/cosmos-db/global-dist-under-the-hood

- Multi-regional accounts
  - Number of regions → multiplier on cost
- Single-master vs. multi-master
  - 100 RU/s for multi-master are twice as expensive as for single-master
- Auto-scale vs. manually provisioning throughput
  - 100/RU/s for Auto-Scale are 1.5 times as expensive as for manual throughput
- Pricing: https://azure.microsoft.com/en-us/pricing/details/cosmos-db/autoscale-provisioned/
- Consistency-level
  - Consistency levels: https://learn.microsoft.com/en-us/azure/cosmos-db/consistency-levels
  - Strong is the only consistency level resulting in synchronous replication
  - Bounded staleness and strong have twice the RU/s charge for reads/queries

How to optimize the data model and query design to reduce RU/s usage?

- Choice of partition/shard key is critical
  - The more unique partition key values the better
  - Avoiding cross partition key queries on the hot path
  - For Cosmos DB: only documents within the same logical partition can be included in multi-document transaction
  - https://learn.microsoft.com/en-us/azure/cosmos-db/partitioning-overview
- Queries
  - Indexing – tune by removing unnecessarily indexed paths and adding useful composite indexes → https://devblogs.microsoft.com/cosmosdb/query-performance-indexing-metrics/
- Wherever possible prefer point-read over query for better throughput/latency/lower RU-charge
- Analytics store (Azure Synapse Link) can be useful for scan-heavy queries/workloads (no RU charge) → https://learn.microsoft.com/en-us/azure/cosmos-db/synapse-link
- Consistency level for individual requests/operations can always be reduced – choose Eventual where functionally possible to reduce the RU-charge